METHOD, SYSTEM, AND PROGRAM FOR HANDLING REDIRECTS IN A SEARCH ENGINE

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0001] The present invention is related to handling redirects in a search engine.

2. Description of the Related Art

[0002] The World Wide Web (also known as WWW or the "Web") is a collection of some Internet servers that support Web pages that may include links to other Web pages. A Uniform Resource Locator (URL) indicates a location of a Web page. Also, each Web page may contain, for example, text, graphics, audio, and/or video content. For example, a first Web page may contain a link to a second Web page.

[0003] A Web browser is a software application that is used to locate and display Web pages. Currently, there are billions of Web pages on the Web.

[0004] Web search engines are used to retrieve Web pages on the Web based on some criteria (e.g., entered via the Web browser). That is, Web search engines are designed to return relevant Web pages given a keyword query. For example, the query "HR" issued against a company intranet search engine is expected to return relevant pages in the intranet that are related to Human Resources (HR). The Web search engine uses indexing techniques that relate search terms (e.g., keywords) to Web pages.

[0005] Some Web pages do not contain content, but, instead, contain a "redirect" to another Web page. For example, if a given Web page A (i.e., a source) redirects to another Web page B (i.e., a target), the Web browser shows Web page B whenever a request for Web page A is received. There are several ways of implementing redirects, including Hyper Text Transfer Protocol (HTTP) redirects (e.g., with HTTP return codes 301 and 302), the use of a META REFRESH tag in Hyper Text Markup Language (HTML), and scripting languages such as JavaScript.

[0006] Redirects are a challenge to Web search engines since the content of a target page should be used to index a source page. For instance, if Web page A redirects to Web page B, then the URL of Web page A should be indexed with the content of Web page B because Web page A has no content, just the redirect (e.g. the JavaScript code that does the redirect). Moreover, redirects may form chains (e.g., Web page A redirects to Web page B, which in turn redirects to Web page C), in which case the transitive closure relationship should be resolved. Additionally, redirect chains may have cycles (e.g., Web page A redirects to Web page B, which redirects to Web page C, which redirects to Web page A), in which case these Web pages should not be indexed because the Web browser cannot display them. Conventional search engines do not handle redirects well. Additionally, conventional search engines handle redirects when "crawling" (i.e., retrieving Web pages), and so they lose the ability to use redirect information in conjunction with, for example, ranking, duplicate detection, and anchor text processing.

[0007] Thus, there is a need for improved redirect processing.

SUMMARY OF THE INVENTION

[0008] Provided are a method, system, and program for handling redirects in documents. At least one equivalence class that includes documents that are connected through a redirect. Cycles for each equivalence class are detected, wherein documents in a cycle are marked so that they are not indexed. Incomplete chains for each equivalence class are detected, wherein documents in an incomplete chain are marked so that they are not indexed. A representative for each equivalence class is selected.

BRIEF DESCRIPTION OF THE DRAWINGS

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 illustrates, in a block diagram, a computing environment in accordance with certain implementations of the invention.

FIGs. 2A and 2B illustrate logic implemented to generate an index in accordance with certain implementations of the invention.

FIG. 3 illustrates logic for performing a document search in accordance with certain implementations of the invention.

FIG. 4 illustrates an architecture of a computer system that may be used in accordance with certain implementations of the invention.

DETAILED DESCRIPTION

[0009] In the following description, reference is made to the accompanying drawings which form a part hereof and which illustrate several implementations of the present invention. It is understood that other implementations may be utilized and structural and operational changes may be made without departing from the scope of the present invention.

[0010] FIG. 1 illustrates, in a block diagram, a computing environment in accordance with certain implementations of the invention. A client computer 100 is connected via a network 190 to a server computer 120. The client computer 100 may comprise any computing device known in the art, such as a server, mainframe, workstation, personal computer, hand held computer, laptop telephony device, network appliance, etc. The network 190 may comprise any type of network, such as, for example, a Storage Area Network (SAN), a Local Area Network (LAN), Wide Area Network (WAN), the Internet, an Intranet, etc. The client computer 100 includes system memory 104, which may be implemented in volatile and/or non-volatile devices. One or more client applications 110 and a viewer application 112 may execute in the system memory 104. The viewer application 112 provides an interface that enables searching of a set of documents (e.g., stored in one or more data stores 170. In certain implementations, the viewer application 112 is a Web browser.

[0011] The server computer 120 includes system memory 122, which may be implemented in volatile and/or non-volatile devices. A search engine 130 executes in the system memory 122. In certain implementations, the search engine includes a crawler component 132, a static rank component 134, a redirect component 136, a duplicate detection component 138, an anchor text component 140, and an indexing component 142. Although components 132, 134, 136, 138, 140, and 142 are illustrated as separate

components, the functionality of components 132, 134, 136, 138, 140, and 142 may be implemented in fewer or more or different components than illustrated. Additionally, the functionality of the components 132, 134, 136, 138, 140, and 142 may be implemented at a Web application server computer or other server computer that is connected to the server computer 120. Additionally, one or more server applications 160 execute in system memory 122.

[0012] The server computer 120 provides the client computer 100 with access to data in at least one data store 170 (e.g., a database). Although a single data store 170 is illustrated, for ease of understanding, data in data store 170 may be stored in data stores at other computers connected to server computer 120.

[0013] Also, an operator console 180 executes one or more applications 182 and is used to access the server computer 120 and the data store 170.

[0014] The data store 170 may comprise an array of storage devices, such as Direct Access Storage Devices (DASDs), Just a Bunch of Disks (JBOD), Redundant Array of Independent Disks (RAID), virtualization device, etc. The data store 170 includes data that is used with certain implementations of the invention.

[0015] FIGs. 2A and 2B illustrate logic implemented to generate an index in accordance with certain implementations of the invention. Control begins at block 200 documents that are to be indexed by the search engine 130 are obtained. In certain implementations, the documents are published or pushed (e.g., as may be the case with newspaper articles) to the indexing component 142. In certain implementations, the crawler component 132 discovers, fetches, and stores the documents. In certain implementations, the crawler component 132 may discover documents based on, for example, certain criteria (e.g., documents were accessed within the last month). Additionally, the crawler component 132 may discover documents in one or more data stores connected directly (e.g., data store 170) or indirectly (e.g., connected to server computer 120 via another computing device (not shown)) to server computer 120. In certain implementations, the crawler component 132 discovers, fetches, and stores Web pages in data store 170. In certain

implementations, the crawler component 132 may associate an indicator with a document if the document contains a redirect.

[0016] In block 202, the static rank component 134 reviews the stored documents and assigns a rank to the documents. The rank may be described as the importance of the source document relative to other documents that have been stored by the crawler component 132. Any type of ranking technique may be used. For example, documents that are accessed more frequently may receive a higher rank.

[0017] In block 204, the redirect component 136 builds a redirect file based on the stored documents. In certain implementations, the redirect component 136 performs a data store scan to read the documents stored by the crawler component 132 and determines whether the document contains a redirect. In certain implementations, the redirect component 136 determines whether a document contains a redirect based on meta-data associated with each document, markup data associated with each document, or content of each document. For instance, the redirect component may detect an HTTP return code (e.g., codes 301 or 302 in a HTTP header) indicating a redirect in the document, a META REFRESH directive in a HTML document, or a script (e.g., JavaScript) that performs a redirect in the document.

[0018] If the document contains a redirect, the redirect component 136 determines a target document of the redirect. In the case that a document contains a redirect , the redirect component 136 stores an entry for the document in the redirect file. In certain implementations, an entry consists of a source path (e.g., URL), a target path (e.g., URL), and a redirect type. A path may be described as data that indicates a location of a document.

[0019] Although examples herein may refer to HTML format or HTTP meta data, certain implementations of the invention are applicable to other formats and other retrieval protocols (e.g., different formats such as XML or Portable Document Format (PDF) and different retrieval protocols such as symbolic links in file systems, the File Transfer Protocol (FTP), etc.).

[0020] In block 206, the redirect component 136 identifies redirect chains in the redirect file. One example of a redirect chain occurs when a first document redirects to a second document, which in turn redirects to a third document. In certain implementations, the redirect component 136 identifies the redirect chains by scanning the redirect file and building a "union find" data structure to identify the redirect chains. The "union find" data structure could, for instance, be a mapping from documents to equivalence classes. An equivalence class may be described as including documents that are connected through a redirect. For example, if a source document has a redirect to a target document, the source document and target document are mapped to the same equivalence class.

[0021] Initially, each document is in its own equivalence class. Then, for each entry in the redirect file, if a first document redirects to a second document, the equivalence classes of the first and second documents are unified. Continuing with this processing, if the second document redirects to a third document, then the third document is in the same equivalence class as the first and second documents. The redirect component 136, thus, processes the entries in the redirect file to identify redirect chains in the form of equivalence classes.

[0022] In block 208, the redirect component 136 detects cycles in the redirect chains. For example, a cycle occurs when a first document redirects to a second document, which redirects to a third document, which redirects back to the first document. Once the redirect chains are identified, the redirect component 136 performs cycle detection. In particular, cycle detection analyzes each redirect chain, looking for cycles. If a cycle is detected in a redirect chain, the redirect component 136 marks the documents involved in that redirect chain with a "do not index" indicator (e.g., flag), which indicates to the indexing component 142 that these documents are invalid documents that should not be indexed.

[0023] In block 210, the redirect component 136 detects incomplete chains in the redirect chains. In certain implementations, the incomplete chain detection is performed for chains that are not marked with a "do no index" indicator. An example of an incomplete

chain occurs when the documents in a single redirect chain are redirects, R1->R2->...->Rn, where Rn is a redirect to a document that was not discovered, fetched, and stored by the crawler component 132 (i.e., "crawled"). This redirect chain is considered incomplete because there is no content associated with Rn (because it was not "crawled"). The redirect component 136 marks documents in the incomplete redirect chain with a "do not index" indicator.

[0024] In block 212, the redirect component 136 selects a representative for each redirect chain, and, if needed, propagates content of a target document with the selected representative. The technique for selecting a representative may be adapted to a search engine 130 policy. For instance, if a first document redirects to a second document, the content of the second document may be indexed with the path of either the first or second document. The selection may depend on the redirect type, for example, whether the redirect is permanent or temporary, or may be based on a static rank computed for each document.

[0025] In certain implementations, the redirect component 136 selects a representative for each redirect chain (e.g., equivalence class) whose documents have not been marked with a "do not index" indicator. In certain implementations, the representative is a path (e.g., a URL) with which the content of the final target document in the chain is indexed. The final target document in the redirect chain contains content, while the other documents in the redirect chain redirect to the final target document. The other documents may directly (e.g., a second document redirects to the final target document) or indirectly (e.g., a first document redirects to a second document that redirects to the final target document) redirect to the final target document.

[0026] In certain implementations, the redirect component 136 selects a representative based on the type of the redirects (e.g., permanent or temporary), a static rank assigned to each document by the static rank component 134, or based on other criteria. For instance, for HTTP permanent redirects, the redirect component 136 may select the path (e.g., URL) of the target document, while for HTTP temporary redirects, the redirect component 136 may select the path (e.g., URL) of a source document. If selection is

based on a ranking, a document with a highest ranking in an equivalence class may be selected. In certain embodiments, the paths for documents that are not selected are marked with an "ignore" indicator so that these paths are not included in an index.

[0027] Moreover, the content of the target document may be propagated to the selected representative. For example, if the selected representative includes a temporary redirect, the content of the target document is propagated to the selected representative.

[0028] In block 214, the search engine 130 determines whether duplication detection is to be performed. If so, processing continues to block 216, otherwise, processing continues to block 218. In block 216, the duplicate detection component 138 detects duplicate documents in different redirect chains and merges the redirect chains. In certain implementations, the duplicate detection component 138 uses a content-based duplicate detection technique that uses information about the documents in the redirect chain (i.e., the equivalence class) in the "union find" data structure. In certain embodiments, two documents may be considered to be duplicates if they are similar (e.g., more than some percentage (e.g., 90%) of their content is the same). For example, if a first and second document are considered equivalent by content, and if a third document redirects to the first document and a fourth document redirects to the second document, the redirect component 136 concludes that the first, second, third, and fourth documents are equivalent. Additionally, if a redirect chain has a first document and another redirect chain has a second document, and if the first document and second document are duplicates based on content, the redirect chains containing the first and second documents are merged to form one redirect chain.

[0029] In block 218, the search engine 130 determines whether anchor text processing is to be performed. If so, processing continues to block 220, otherwise, processing continues to block 222. In block 220, the anchor text component 140 performs anchor text processing. Anchor text may be described as text associated with a path or link (e.g., a URL) to a document. In certain implementations, anchor text is text that labels or encloses hypertext text links in Web documents.

[0030] The search engine 130 may collect, for each document, the anchor text of paths that point to that document, and then the anchor text may be indexed along with the document content. For example, if a first document links to a second document, and the second document is a redirect to a third document, then the anchor text of the path from the first document to the second document is indexed along with the content of the third document, if the document is considered the representative of the redirect chain.

[0031] In certain implementations, the anchor text component 140 uses an anchor text indexing technique that uses information about the documents in the redirect chain (i.e., the equivalence class) in the "union find" data structure to propagate anchor text to the representatives of redirect chains. As another example, if a first and a second document are in a redirect chain (e.g., an equivalence class) whose representative is a third document, then the anchor text pointing to the first document and to the second document is indexed for the third document, along with the content of the target document of the redirect.

[0032] In block 222, the indexing component 142 generates an index. In particular, for each redirect equivalence class, the indexing component 142 locates the target document that contains content (e.g., the target of the redirects) and indexes that content with the path (e.g., URL) of the representative for the redirect chain (e.g., equivalence class). The indexing component 142 performs indexing for the documents stored by the crawler component 132.

[0033] In certain implementations, paths of each document in the equivalence class may also be propagated to the selected representative for global analysis, which is described further in United States Patent Application No. xx/xxx,xxx, entitled "A PIPELINED ARCHITECTURE FOR GLOBAL ANALYSIS AND INDEX BUILDING," by Marcus F. Fontoura et al., Docket No. SVL920030120US1, filed on the same date herewith, and which is incorporated by reference herein in its entirety.

[0034] FIG. 3 illustrates logic for performing a document search in accordance with certain implementations of the invention. Control begins at block 300 with a user submitting a search request via the viewer application 112. In block 302, the search

engine 130 executes the search request. In block 304, the search engine returns search results that include the redirect and other processing described in FIGs. 2A and 2B. In block 306, the viewer application 112 displays the search results.

[0035] Thus, certain implementations of the invention address the problem of indexing redirects in a search engine 130. Certain implementations of the invention identify redirects and group documents into equivalence classes. Certain implementations of the invention also identify cycles in redirect chains and incomplete redirect chains and avoid indexing the documents in a cyclic or incomplete redirect chains (i.e,. the documents marked "do not index" are not included in the index generated by the indexing component 142). Also, for each redirect chain, certain implementations of the invention identify a path for which the content is to be indexed, and marks all other paths to be ignored. Certain implementations of the invention work in conjunction with a content-based duplicate detection technique and with an anchor text indexing technique. The result of the processing of certain implementations of the invention is an improved index. Thus, the quality of results delivered by the search engine to its users is improved.

Additional Implementation Details

[0036] The described techniques for handling redirects in a search engine may be implemented as a method, apparatus or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The term "article of manufacture" as used herein refers to code or logic implemented in hardware logic (e.g., an integrated circuit chip, Programmable Gate Array (PGA), Application Specific Integrated Circuit (ASIC), etc.) or a computer readable medium, such as magnetic storage medium (e.g., hard disk drives, floppy disks,, tape, etc.), optical storage (CD-ROMs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, firmware, programmable logic, etc.). Code in the computer readable medium is accessed and executed by a processor. The code in which various implementations are implemented

may further be accessible through a transmission media or from a file server over a network. In such cases, the article of manufacture in which the code is implemented may comprise a transmission media, such as a network transmission line, wireless transmission media, signals propagating through space, radio waves, infrared signals, etc. Thus, the "article of manufacture" may comprise the medium in which the code is embodied. Additionally, the "article of manufacture" may comprise a combination of hardware and software components in which the code is embodied, processed, and executed. Of course, those skilled in the art will recognize that many modifications may be made to this configuration without departing from the scope of the present invention, and that the article of manufacture may comprise any information bearing medium known in the art.

[0037] The logic of FIGs. 2A, 2B, and 3 describes specific operations occurring in a particular order. In alternative implementations, certain of the logic operations may be performed in a different order, modified or removed. Moreover, operations may be added to the above described logic and still conform to the described implementations. Further, operations described herein may occur sequentially or certain operations may be processed in parallel, or operations described as performed by a single process may be performed by distributed processes.

[0038] The illustrated logic of FIGs. 2A, 2B, and 3 may be implemented in software, hardware, programmable and non-programmable gate array logic or in some combination of hardware, software, or gate array logic.

[0039] FIG. 4 illustrates an architecture of a computer system that may be used in accordance with certain implementations of the invention. For example, client computer 100, server computer 120, and/or operator console 180 may implement computer architecture 400. The computer architecture 400 may implement a processor 402 (e.g., a microprocessor), a memory 404 (e.g., a volatile memory device), and storage 410 (e.g., a non-volatile storage area, such as magnetic disk drives, optical disk drives, a tape drive, etc.). An operating system 405 may execute in memory 404. The storage 410 may comprise an internal storage device or an attached or network accessible storage.

Computer programs 406 in storage 410 may be loaded into the memory 404 and executed by the processor 402 in a manner known in the art. The architecture further includes a network card 408 to enable communication with a network. An input device 412 is used to provide user input to the processor 402, and may include a keyboard, mouse, pen-stylus, microphone, touch sensitive display screen, or any other activation or input mechanism known in the art. An output device 414 is capable of rendering information from the processor 402, or other component, such as a display monitor, printer, storage, etc. The computer architecture 400 of the computer systems may include fewer components than illustrated, additional components not illustrated herein, or some combination of the components illustrated and additional components.

[0040] The computer architecture 400 may comprise any computing device known in the art, such as a mainframe, server, personal computer, workstation, laptop, handheld computer, telephony device, network appliance, virtualization device, storage controller, etc. Any processor 402 and operating system 405 known in the art may be used.

[0041] The foregoing description of implementations of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many implementations of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.